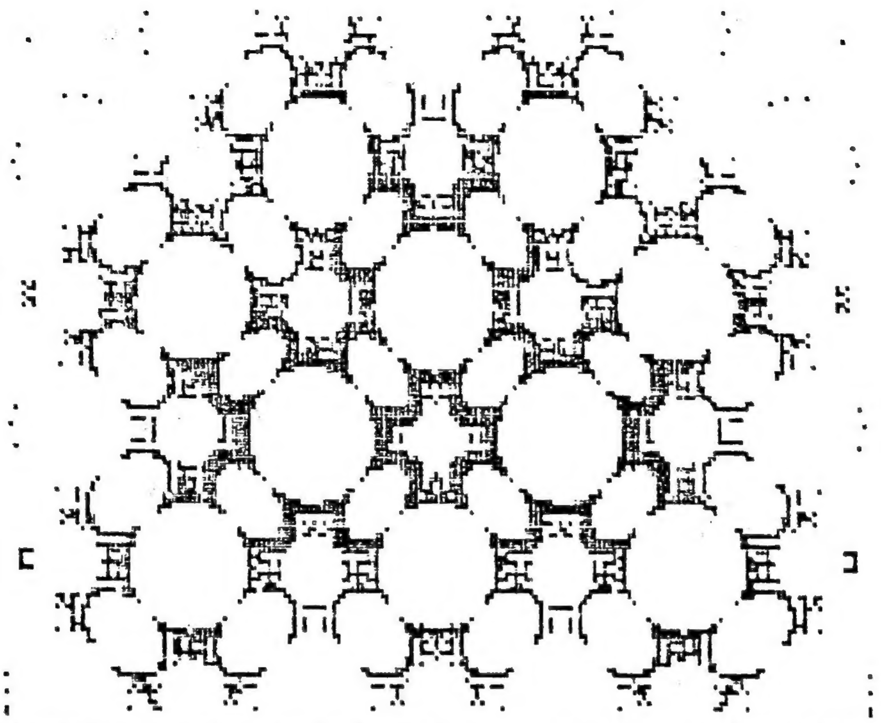
BETA BASIC NEWSLETTER No. 14


Another strange pattern - handy for telling the issues apart!

****

FASTER "XYLEM" (See Newsletter 13)

G. Jackson of Cardiff writes:

"The program is a more-machine-code version of Mr. Thomson's 'XYLEM' in Newsletter 13. It permits the entry of just one parameter instead of XRG, XOS, YRG and YOS and it about doubles the speed of working. Some of Mr. Thomson's fast square root routine is used at L200, and the method of stacking x,y,p,q,r at L100, 140 is the outcome of correspondence with him. There being no end to the modifying of other people's programs I can understand that you may be reluctant to publish this."

Readers are often stimulated to improve on something they see in the Newsletter, which I see as part of its job. It's true I generally don't publish minor improvements, but in this case I think the speed improvement makes it worthwhile. The PRINT £3 in line 110 caused the program to halt with my set-up, but altering it to just PRINT cured the problem, apart from displaying an unwanted number on the screen. Change the pattern by adding or deleting REMs from lines 30 to 60.

```
   10 REM "XYLEM" in m/code (see Newsletter no.13)
      Uses f, with r, to set xrg,xos,yrg,yos
   20 ON ERROR
        IF error=11 THEN RETURN
        ELSE POP
          CONTINUE
   30 REM INPUT "p= ";p'"q= ";q'"r= ";r'"scale factor=";f
   40 REM LET p=.21,q=-20,r=-100,f=4.5
   50 REM LET p=.3,q=-17,r=SQR 2,f=20
   60 LET p=.2,q=-4,r=-3,f=50

   70 PRINT #0;"p,q,r=";p;",";q;",";r;": f=";f
```

```
 80 RESTORE 140
    LET n=0
    DO
      READ a
    EXIT IF a<0
      POKE  23300+n,a
      LET n=n+1
    LOOP
 90 LET xrg=2e3/f,xos=xrg/2-r, yrg=0.69*xrg,yos=yrg/2,x=0,
    y=0
100 REM Run USR . (y ON stack)
110 PRINT #3;y AND USR 23300,x,p,q,r
120 INPUT ;
    PRINT BRIGHT 1;"finished";#0;"p,q,r=";p;",";q;",";r;": f
    =";f
    PAUSE 0
    STOP

130 REM Stack x,p,q,r.Store r,q,p,x,y in mems4-0, and no.of
    plots in bc.
140 DATA 205,121,28,205,121,28,239,196,2,195,2,194,2,193,2,1
    92,2,56,1,0,60
150 REM test for BREAK,push bc
160 DATA 205,84,31,208,197
170 REM ABS(p*x-q)to mem5
180 DATA 239,224,225,226,4,227,3,42,61,197,56
190 REM calc SQR.(to mem5)
200 DATA 126,167,40,19,198,128,31,119,35,54,127,6,2,239,49,2
    29,1,5,15,56,53,16,246
210 REM calc z (see Newsletter 13)
220 DATA 239,225,41,4,3
230 REM y=r-x TO mem0.
240 DATA 228,225,3,192
250 REM x=z to mem1.
260 DATA 1,193
270 REM stack y+x,y-x
280 DATA 15,224,225,3,56
290 REM plot y+x,y-x
    pop bc
    dec bc
    jpnz L160
    ret
300 DATA 205,79,220,193,11,120,177,194,25,91,201,-1
```

****

SAM NEWS

Word of MGT's new computer was a bit slow to reach some parts of Europe - I confused one correspondant into thinking, what with my recent marriage, that Sam was my son! Two slight errors in the item in issue 13 - the price should be in pounds, not hashes, and the DRAW speed is about 7 times faster than a Spectrums, not 5 times. (CIRCLE is at least 50 times faster. I must say I am getting addicted to watching ever-changing colourful graphics demos - how did I ever cope without pixel-level colour?) Unfortunately there have been some delays getting the required custom chip into production, so the machine will not be ready till about September. The late publication of this Newsletter, and my slow response to letters, is largely because I have been very busy completing some sections of SAM's Basic - sorry about that.

****
CAT_TO for the PLUS D

The procedure CAT_TO in issue 9, which catalogues discs drives or Microdrives to a string, generated quite a bit of favourable comment. I also received several applications of the procedure to deal with the PLUS D's catalogue in various ways. Below is a nice program from David Oliver (Houghton-le-Spring, Co. Durham) that prints an alphabetically sorted PLUS D catalogue. He says:

"Loc in line 40 is set to 35 because that is the location of the immediate carriage return code before the first PLUS D file name (i.e. 1 +SYS... etc.). Lines 50 to 80 look for the last carriage return code immediately after the last disc file. This I found, is always the fourth and is used for ending the search and joining to the array e$ in lines 90 to 160. The report giving the number of free bytes is printed by the slicer in line 230 which I found is always the same. If the number of files on disc exceed that which can normally be displayed on one screen then the use of CSIZE 4,8 comes in handy. You obtain a four column display when the comma is used after print in line 210. The program took about 16 seconds to display 80 sorted names."

David also enclosed a version that printed just files of a particular type, using type data he had included as part of the file names, but I have not got room for it. I have not listed the CAT_TO procedure again, either - you will have to look it up in issue 9! It is only 8 lines long.

```
 10 DIM e$(1,10)
 20 cat_to c$
 30 CSIZE 4,8
 40 LET loc=35,d$=CHR$ 13,x=0
 50 FOR a=LEN c$ TO 0 STEP -1
 60   IF CODE c$(a)=13 THEN LET x=x+1
 70   IF x=4 THEN LET end=a
        GO TO 90
 80 NEXT a
 90 DO
100   LET loc=INSTRING(loc,c$,d$)
110 EXIT IF loc=end
120   DIM b$(1,10)
130   LET b$(1)=c$(loc+4 TO loc+13)
140   JOIN b$ TO e$
150   LET loc=loc+1
160 LOOP
170 IF e$(LENGTH(1,"e$"),1)=" " THEN
      PRINT "Thats all folks!"
      CSIZE 0
      STOP
180 DELETE e$(1)
190 SORT e$()
200 FOR b=1 TO LENGTH(1,"e$")
210   PRINT e$(b),
220 NEXT b
230 PRINT c$(LEN c$-29 TO LEN c$-1)
240 CSIZE 0
    STOP
```

Note the way David DIMed an array big enough for just ONE name to start with, adding new names and expanding the array using JOIN only as needed. The first, blank, entry in the array is in fact deleted before SORT is used.

****

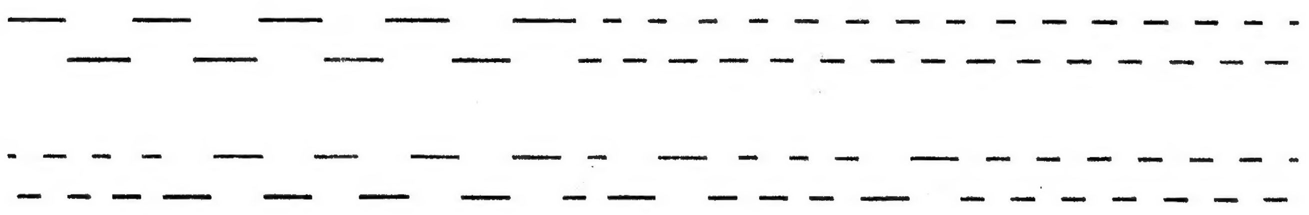
DIGITAL OSCILLOSCOPE

An oscilloscope is one of those display devices you see next to the bedside when a character in a soap opera has been shot. They are also used by all sorts of technical and scientific people in order to look at fast changing signals (and to save paper!). The program below does not use any BB features. However, I have been asked to show how the EAR socket can be used for data input, and the machine code routine provided could form the basis for further experimentation. It checks to see if the value at the EAR socket is high or low and plots the results a screenful at a time. The sampling speed can be altered by changing the value of the *speed* variable. A value of 1 gives the fastest speed; higher numbers give slower displays - not exactly proportional to *speed*, though. Unlike most oscilloscopes, no vertical lines are drawn - but then the EAR socket does not allow values other than "high" or "low".

The illustration shows part of a tape header, simply played while the program was running. The start of the top line is the end of the leader, made up of pulses about 1.2msecs. long (per complete up/down cycle). Then we can see the start of the information carried by the header; the first narrow pulse is a synch pulse, then binary zeros are shown by narrow pulses, and ones by wide pulses. With a bit of squinting, you should be able to read "00000000 00000000 01111010 00100000" or 0,0,122,32 in decimal. The gaps are mine - the Spectrum tape system has no gaps between bytes. The first zero means "header" rather than "data" (a main block of data is preceded by 255), the next zero shows the type is "program", and 122 and 32 are the character codes for "z" and " ", the start of the name (which was "z"). Bye the way, I am *not* putting this forward as a header reading program - just as an example!

I have used this "oscilloscope" on voice recordings, and (cautiously) on same of the signals inside the computer. The latter were generally too fast to show much except that a signal was present. It should work on Centronics signals. I am hesitant to try RS232 signals, because I am not sure of the maximum voltages it is safe to feed into the EAR socket. Note: The EAR socket will register an *unchanging* high voltage as "low".

```
 10 LET crt=USR "a"
 20 FOR n=crt TO crt+44
 30 READ a: POKE n,a: NEXT n
 40 LET speed=5: POKE crt+30,speed
 50 RANDOMIZE USR crt
 60 PAUSE 25: CLS: GO TO 50
100 DATA 243,175,79,245,205,176,34,92,22,128,65,219,254
110 DATA 230,64,123,32,2,198,7,103,122,182,119,203,10,48
120 DATA 1,44,14,16,13,32,253,16,231,241,198,24,254,169
130 DATA 56,216,251,201
```



****
RECORDING SOUNDS IN MEMORY

Another use of the EAR socket is recording sounds in memory and playing them back. PROC REC records anything at the EAR socket into an array. An optional parameter controls the sampling speed, allowing you to trade off sound quality against memory usage. PROC PLY will play your arrays back, also at a controllable speed; if this is the same as the recording speed, the played sound will be normal(ish); alternatively, faster or slower speeds can be used. (Small values of T give high speeds.)

Prepare a tape of the sound you want to record, run the program, then play the tape into the EAR socket. I was startled to find that my voice played at a higher speed sounds rather like my sister's! I used the procedures as the basis for a "speaking clock" program which sounded pretty horrible but was quite entertaining. I also tried with some success to remove some of the high-frequency crackle from sounds by processing the array data in various ways. A graphical version of the arrays can be obtained by using a loop to POKE them onto the screen 32 characters at a time, then scroll the screen by one pixel. I found this useful for looking at crackle and hiss.

```
 10 setup
 20 DIM p$(15000)
 30 PRINT "Any key to record"
 40 PAUSE 0
 45 PRINT "recording..."
 50 rec p$
 60 PRINT "Any key to play"
 70 PAUSE 0
 75 FOR t=1 TO 16 STEP 3
 80   ply p$,t
 90 NEXT t

100 DEF PROC setup
110   LET ad=USR "a"
120   IF PEEK ad<>33 THEN
        FOR n=ad TO ad+98
          READ a
          POKE n,a
        NEXT n
130 END PROC

140 DEF PROC rec REF a$,tim
145   DEFAULT tim=8
150   DPOKE ad+1,LENGTH(0,"a$")
160   DPOKE ad+4,LENGTH(1,"a$")
165   POKE ad+7,tim
170   RANDOMIZE USR ad
180 END PROC

190 DEF PROC ply REF a$,tim
195   DEFAULT tim=8
200   DPOKE ad+43,LENGTH(0,"a$")
210   DPOKE ad+46,LENGTH(1,"a$")
215   POKE ad+50,tim
220   RANDOMIZE USR (ad+42)
230 END PROC
```

```
    500 DATA 33,0,0,17,0,0,14,0,243,24,20,227,227,0,219,254,219,
        254,65,0,0,16,254,23,23,203,22,48,238,35,27,122,179,32,
        2,251,201,0,54,1,24,230
    510 DATA 33,0,0,17,0,0,217,14,0,217,58,72,92,31,31,31,230,7,
        71,14,254,243,24,24,227,227,0,56,2,203,160,48,2,203,224,
        217,65,1,6,254,217,237,65,203,39,32,234,35,27,122,179,
        126,55,23,32,228,251,201
```

Note: If you add up all the numbers in these horrible DATA statements the total should be 9691 or one of us has made a mistake...

****
PROC GRID - highlighting attribute boundaries

Antony Legat of Blakedown, nr. Kidderminster, Worcs. writes:

"PROC GRID is a simple procedure to provide a grid overlay, highlighting attribute boundaries and displaying line and column numbers (CSIZE 0). I have found this to be a genuinely useful procedure, and just the type of thing Newsletters should be full of!!"

```
   9000 DEF PROC GRID
          LOCAL L,C,A$
          LET A$=STRING$(16,CHR$ 104+CHR$ 40)+STRING$(16,CHR$ 40
          +CHR$ 104),A$=STRING$(11,A$)
          FOR L=0 TO 21
            PRINT CSIZE 4,8;AT L,0;L;" "
          NEXT L
          FOR C=2 TO 30 STEP 2
            PRINT INVERSE 1; CSIZE 4,8;AT 0,C*2;C
          NEXT C
          POKE 22528,A$
        END PROC
```

Antony has used the fast method of preparing a string containing attribute codes for a chess-board of BRIGHT/not BRIGHT character squares. This is POKEd to the attributes area at the end of the procedure. (CHR$ 40 is 00101000 in binary - which the Spectrum displays as "not FLASH" (first 0), "not BRIGHT" (next 0), PAPER 5 (the "101") and INK 0 (the "000"). CHR$ 104 is 01101000 - the same, but with the BRIGHT bit set to 1. Those who, like me, use a monitor that does not respond to BRIGHT can modify the PROC to use PAPER variations instead.

****
PROC FIND - finding the coordinates of a pixel

This procedure is the second one from Antony Legat. I have found myself PLOTing by trial and error in order to find the right place to start a FILL from; the procedure is designed to remove the need for such guessing, in graphics work, or when placing characters precisely with PLOT. It lets you move a pixel cursor while displaying its coordinates.

The cursor keys referred to are 5,6,7 and 8. I found the cursor speed a little slow; if you agree, you could consider allowing other keys to move the cursor in bigger steps, or perhaps printing X and Y only once, on leaving the procedure. (This is when you usually need to know the coordinates, after all.)

```
9000 DEF PROC FIND
9010   LOCAL X,Y,K
9020   BORDER 1
9030   INK 9
9040   INPUT ;
9050   LET X=128,Y=88
9060   PLOT OVER 1;X,Y
9070   PRINT #0;AT 0,0;"X=";X;" ","Y=";Y;" "' INK 5;"CURSOR K
       EYS-MOVE, 9-EXIT/CLEAR"
9080   DO
9090     GET K
9100     PLOT OVER 1;X,Y
9110     ON K-4
           LET X=X-1
           LET Y=Y-1
           LET Y=Y+1
           LET X=X+1
9120     LET X=X-(X>255)+(X<0)
9130     LET Y=Y-(Y>175)+(Y<0)
9140     PLOT OVER 1;X,Y
9150     PRINT #0;AT 0,0;"X=";X;" ","Y=";Y;" "
9160   LOOP UNTIL K=9
9170   POKE 23624,PEEK 23693
9180 END PROC
```

Antony changed the border colour to show up the screen edge. Line 9170, which resets the border, is interesting; it POKEs the border colour system variable with the permanent upper screen colours (set by e.g. PAPER 2). The border doesn't actually change until the next keypress, when the associated click requires the ROM to send the value in BORDCR to the port used to control both sound and border. The border will always match the paper colour of the main screen, with this method. An alternative approach would be to find the border colour before altering it and then assign it to a variable, so that it could be reset using BORDER at line 9170. You could use somthing like:

   LET bd=INT(PEEK 23624/8) to find the current border colour.

****
BB 4.0 - RAMDISC ARRAY SYNTAX WHEN THE NAME IS A VARIABLE.

H.N.S. Wijegoonawardena (Edgware, Middx.) sent this useful summary of the more indirect ways of using Ramdisc arrays in BB 4.0 - using a VARIABLE for the array name. (For example, using DIM !b$,(20,10) rather than DIM !test$(20,10).)

```
STRING ARRAYS:     LET b$="test$"
                   DIM !b$,(20,10)
                   LET !b$,(4)="zaza"
                   PRINT !b$,(4)
                   PRINT LENGTH(2,"!b$,()") is equivalent to
                   PRINT LENGTH(2,"!test$()")
                   SORT !b$,()
                   LIST !b$

NUMERIC ARRAYS:    LET b$="num"
                   DIM !b$,(10,5)
                   LET !b$,(1,5)=78
                   PRINT !b$,(1,5)
                   PRINT LENGTH(2,"!b$,()") is equivalent to
                   PRINT LENGTH(2,"!num()")
                   LIST !b$
```

****

PROC SETUP - setting up tables

Bill Pedder of Hemel Hempstead, Herts., sent me a program for setting up table data. The original subroutine for entering numbers was rather long, and I have substituted a shorter version to save space (and your typing!). Bill talks about rows and row length where I would have said columns and column width - this might be confusing. He writes:

"In programming I always use BB, the programs are mainly for calculations in structural engineering. Referring back to earlier issues of the Newsletter and INPUT procedures, I find a requirement is often for a table of titled lines and rows with data placed in the appropriate lines and rows. I enclose PROC SETUP on the enclosed tape. It calls on:

PROC BD	- To draw a border round the screen
PROC TABLE	- To define the number of lines and rows
PROC TITLEL	- To title the lines
PROC TITLER	- To title the rows
PROC I	- To put in the data
PROC C	- To copy screen to printer (Alphacom in my case)

The first INPUT required is FORMAT. This is put in as ####.# or whatever is required. The length of each row is then fixed to the number of characters (6 in this case). Titles are then given to the lines and rows. Data is then put in the appropriate line, row. The particular program for calculations required is then written and the results put in the appropriate line,row in the table. When putting DATA in a particular line or row if OUTPUT is required at this location then zero is INPUT to be overwritten later when calculations have been made. The subroutine at line 9072 is to receive only an INPUT number with or without a minus sign or decimal point. I can't seem to get this to work as a PROC.

The things I feel could be improved are at least:
a) Change of subroutine for input a number to a PROC.
b) Changing length of each row as required i.e. ROW 1 "###", ROW 2 "##.##", ROW 3 "##.#" etc. "

Editor's comments: I wrote a quite similar progam when I did medical research. Given such a program as a basis, it is fairly easy to add sections to calculate averages, add, subtract, multiply or whatever one column by another, and perform statistical tests. Arrays are good ways of handling lots of numbers. Regarding improvements, I agree about the input routine. A string array could hold format strings for each column separately. Heading strings could be printed centrally by printing TAB (desired column centre) minus half heading length (excluding spaces).

```
   10 LIST FORMAT 2
      CSIZE 4,8
      POKE 23658,8
      BORDER 5
      PAPER 7
      INK 0
      PROC SETUP
```

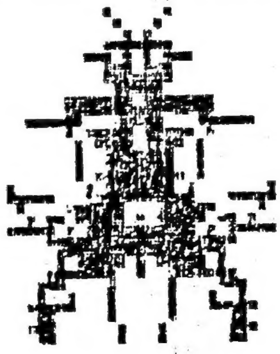
```
9000 DEF PROC SETUP
9002   PROC TABLE
9004   FOR K=1 TO NL
         PROC TITLEL K
       NEXT K
9006   INPUT "LINE TITLES OK?";Q$
       IF Q$="N" THEN INPUT "LINE LETTER TO ALTER?";A$
         LET K=CODE A$-64
         PROC TITLEL K
         GO TO 9006
9008   FOR K=1 TO NR
         PROC TITLER K
       NEXT K
9010   INPUT "ROW TITLES OK?";Q$
       IF Q$="N" THEN INPUT "ROW NUMBER TO ALTER?";A$
         LET K=VAL A$
         PROC TITLER K
         GO TO 9010
9012   FOR K=1 TO NR
         FOR J=1 TO NL
           PROC I J,K
           LET X(J,K)=NU
         NEXT J
       NEXT K
9014   INPUT "DATA OK?";Q$
       IF Q$="N" THEN INPUT "ITEM LETTER TO ALTER?";A$
         LET J=CODE A$-64
         INPUT "ROW NUMBER TO ALTER?";K
         PROC I J,K
         LET X(J,K)=NU
         GO TO 9014
9016   PROC C
9018 END PROC
9020 STOP

9024 DEF PROC TABLE
       LOCAL K
       INPUT "FORMAT? (####.##)";U$
       LET LE=LEN U$
9026   INPUT "NO OF LINES? (18 MAX)";NL
       IF NL<1 OR NL>18 THEN GO TO 9026
         DIM L$(NL,9)
9028   INPUT "NO OF ROWS? ";NR
       IF NR<1 OR NR>INT (48/(LE+1)) THEN GO TO 9028
9030   CLS
       PROC BD
       PRINT AT 1,12; INVERSE 1;"ROW ";AT 2,1; INVERSE 1;"LIN
       E "
9032   FOR K=1 TO NL
         PRINT AT K+2,1;CHR$ (K+64);")"
       NEXT K
       FOR K=1 TO NR
         PRINT AT 1,15+(K-1)*(LE+1)+LE/2;K;")"
       NEXT K
9034   DIM L$(NL,9)
       DIM R$(NR,LE)
       DIM X(NL,NR)
       DIM Z$(LE)
9036 END PROC

9038 REM TITLE LINE
9040 DEF PROC TITLEL K
9042   PRINT AT 2+K,5;"          ";AT 2+K,5; FLASH 1;"?"
       INPUT L$(K)
       PRINT AT 2+K,5;L$(K)
9044 END PROC
```

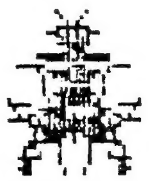
```
9046 REM TITLE ROW
9048 DEF PROC TITLER K
9050   PRINT AT 2,16+(K-1)*(LE+1);Z$;AT 2,16+(K-1)*(LE+1); FL
       ASH 1;"?"
       INPUT R$(K)
       PRINT AT 2,16+(K-1)*(LE+1);R$(K)
9052 END PROC

9054 REM INPUT DATA
9056 DEF PROC I J,K
9058   PRINT AT J+2,16+(K-1)*(LEN U$+1);Z$;AT J+2,16+(K-1)*(L
       EN U$+1); FLASH 1;"?"
       GO SUB 9072
       PRINT AT J+2,16+(K-1)*(LEN U$+1); USING U$;NU
9060 END PROC
9062 STOP

9064 REM BORDER
9066 DEF PROC BD
9068   PLOT 0,0
       DRAW 255,0
       DRAW 0,175
       DRAW -255,0
       DRAW 0,-175
       PLOT 1,1
       DRAW 253,0
       DRAW 0,173
       DRAW -253,0
       DRAW 0,-173
9070 END PROC

9072 REM NUMBER INPUT
9074 ON ERROR 9100
9076 LET ERROR=0
9077 INPUT N$
9078 LET NU=VAL N$
9080 IF ERROR<>0 THEN GO TO 9074
9082 ON ERROR 0
       RETURN

9100 BEEP 1,1
     RETURN

9120 REM COPY
9122 DEF PROC C
       INPUT "COPY? (Y/N)";Q$
       IF Q$="Y" THEN COPY
9124 END PROC
```

****

ANOTHER BB BUG?

Way back in issue 6 I published an item on simulating evolution. Les Tyler (Cheltenham) who has filled many an idle moment running it, submitted the following gene values for a rather more convincing fly: -2, 2, 4, -1, 3, -9, 6, 0, 6.





****
PROC REPEAT (MK. 2)

Here is a compact and devious procedure for repeating a number of statements in a simple manner, sent in by Lars Hult of Goteborg, Sweden. Another REPEAT procedure of a different kind was published in issue 5, page 8, but I cannot think of a sensible new name for this one! The procedure uses XOS as a variable, which is a bit naughty; although XOS does end up with the default value of zero, REPEATed graphics commands will behave oddly. Put the REPEAT at the end of the line you want to repeat a given number of times, e.g:

```
    10 BEEP .1,0
       BEEP .2,5
       repeat 5

  1000 DEF PROC repeat r1
         POP r2
         LET xos=xos+1
         LET xos=xos*(r1<>xos)
         GO TO r2+NOT xos
       END PROC
```

****
SOLVING EQUATIONS

Jan Moller (Lulea, Sweden) writes:
"Here is a simple procedure for solving equations. It uses the algorithm by Newton-Raphson. The equation is entered as a string and used as a value parameter in the procedure. Your guess is entered as a number and sent as a parameter by reference. The procedure prints each new 'x' and the value of the equation for this 'x'. When the difference between two 'x's is smaller than 0.000001 the loop ends and the root and the value of the function are printed."

"To stop unending loops a variable 'count' is updated and after 20 loops leaves. (Sometimes a root may not be found.) There is no check that 'der' <> zero which of course would give a divide by zero error.. 'Der' will be zero at the root if the equation has a double root like x*x*x-8*x*x+20*x-16."

I took the liberty of adding line 130 so that things like SIN and TAN can be entered in spelled-out form. An example: enter the equation x*x-tan x, guess -2 or -3, and the final x will be about -1.85393. (If all this confuses you, don't worry - I only vaguely understand it myself!)

```
   100 INPUT "Eq:";g$
   120 INPUT "Guess x:";xx
   130 LET g$=SHIFT$(8,g$)
   140 EqSolv g$,xx

  9000 DEF PROC EqSolv f$, REF x
         LOCAL h,f1,f2,der,count
  9010   LET h=.00001,count=1
  9040   DO
           LET f2=VAL f$,x=x+h,f1=VAL f$,x=x-h,der=(f1-f2)/h
  9045     LET x=x-VAL f$/der
           PRINT count;TAB 3;x;TAB 18;VAL f$
           LET count=count+1
         LOOP UNTIL ABS (VAL f$/der)<1e-6 OR count>20
         PRINT BRIGHT 1;x,VAL f$
  9050 END PROC
```

****

READERS' LETTERS .

Dear Andy,

Unfortunately, I've been able to get no information on the hardware of the Interface 1, and I wonder how it works. I seem to remember hearing about a 'Shadow ROM', but am not sure what this means.

Antony Legat, Blakedown, Worcs.

*A Z80 chip like the Spectrum's can only deal with 64K at a time. In order for peripherals like Interface 1 to add new commands, the usual system is this: the circuitry in the peripheral is activated when the Spectrum executes the part of ROM at location 8, which happens when a syntax error is detected. This circuitry sends a signal into the edge connector which turns off the internal ROM, and a second (so-called 'Shadow' ROM) in the peripheral is turned on and takes over. The bottom 16K of the Z80 chip's memory is suddenly different, and hey presto, a different ROM is now in control, running the part of its program just after location 8. The 'Shadow' ROM can now find out what caused the error that activated it, and if needed, control e.g. a Microdrive SAVE, before telling the circuitry to switch back to the normal ROM. To examine the Interface 1 ROM, just SAVE *"m";1;"name" CODE 0,16384. This saves the bottom 16K of memory, where the 'Shadow' ROM must be switched in to control the SAVE. Then something like CLEAR 32767: LOAD *"m";1;"name" CODE 32768 will load the code back where you can examine it. I picked 32768 because that is 8000H; a disassembly will look reasonable - e.g. JR 0123H will be listed as JR 8123H. (I used to have a disassembler that would pretend the code it was working on was anywhere you wanted, but that was on a different machine, alas!)*

Dear Andy,

...I was surprised you recommended Format in issue 13 instead of the Spectrum Discovery Club. I joined the SDC after you mentioned it in Newsletter 8 and I have thoroughly enjoyed both the disc-based newsletters and the library program discs. The one issue of Format I've seen was rather poor in comparison.

I found your article on the SAM computer very enlightening and it occured to me that if BB programs and procedures are going to be (almost) compatible might it not be a good idea to process the BB newsletters into a Sam newsletter? This might enable you to earn a little more for your efforts and give Sam users a very good grounding in their Basic. (If it helps to keep the BB newsletter going a bit longer then it can't be bad!)

Paul Hammond, Grimsby, Humberside

I have only recently been made a member of the Spectrum Discovery Club and got some disc back-issues. I agree with you - they are very good! It is particularly convenient to have programs ready to run on the disc, as well as all the other newsletter sections. Their prices are very reasonable (they are non-profit making) at £1.50 per issue or £8.00 for 6 issues; many people would pay more for the discs alone. Quite a large number of people run various sub-sections, so there is a big pool of expertise. (I noticed the names of quite a few BB *users.) In short, if you have an Opus Discovery, you would be mad not to join! (Unless perhaps spelling mistakes cause your blood pressure to rise dangerously....) A better address than the one in Newsletter 8 is:*

B. MUMFORD, 57 ST. SAVIOURS ROAD, WEST CROYDON, SURREY, CR0 2XE

Cheques and Postal Orders should be made out to SDC. Also mentioned in BB Newsletter 8 was Dave Corney, writer of the Opus Discovery ROM. Unfortunately, he seems to have vanished, and some readers have lost money. The ROMs can now be obtained from SDC, who are keeping a special bank account to turn over to Mr. Corney in case he ever shows up. (In which case I hope he settles any debts!)

Another disc/Microdrive based magazine is OUTLET. I have not seen it yet, but reader Scott Brown writes "..how well it works. They get some brilliant utility software sent in." and John Luby wrote "It's one of the few publications I look forward to anymore... I can recommend it to BBN readers. If nothing else it's enthusiastic and cheap, and there have been a few very good utilities in it."

(Rambling letter reply, this, isn't it?) Paul Hammond's idea idea about the BB Newsletter is something I will think about.

Dear Dr. Wright,

Is it possible to define UDGs in CSIZEs other than the standard 8*8 pixel character size? Such a technique would allow me to display large circuit diagrams on screen by using graphics symbols defined in "small" character squares - e.g. 4*4, before "dumping" to a printer.

David Griffiths, Dinas Powys, S. Glamorgan

*Such symbols are probably best done using UDGs or characters plotted in OVER 2. The large border of unused pixels in a standard UDG will then have no effect. The only slight problem might be when you want to overprint symbols, since OVER 2 only adds pixels. Maybe you could use OVER 2 with an all-ink 4*4 symbol, then OVER 1 with the same UDG, to force a 4*4 PAPER "hole"? Then any other symbol could be plotted over the "hole", using OVER 1 again.*

Dear Andy,

I would like to ask if it may be possible to use JOIN / DELETE / COPY commands on RAM DISC arrays... I tried entering some program statements along this line, they passed syntax but stopped with error reports i.e. Invalid argument. My guess to your reply is no.

David Oliver, Houghton-le-Spring, Co. Durham

Correct! In fact, those commands shouldn't get past the syntax check - an error on my part. (Trying to use them can even cause a crash - so don't try!) Because of the way the paged memory works with Ramdisc, the commands would need to be completely re-written and extended to work as we would all like.

Dear Dr. Andy,

No doubt many besides me will be sorry if you have to stop publishing the BB newsletter, but as I hope to be able to transfer to a SAM micro, I shall still feel your influence. ...I have been trying to SORT a PLIST of all the DEF PROCs in a program, and could not. I also have a small club membership list in which each member is noted by name, address and phone number on one line. I think I need to retype it with BB loaded, but so far my experiments have failed to produce anything which will SORT.

Peter Bell, Shoreham-by-Sea, W. Sussex

SORT will only work with arrays - if you want to SORT a set of PROC names you will have to get them into a string array first! If you are using PROC PLIST from issue 4, modify it as follows:

```
   Add: 15 DIM a$(50,20): LET p=1
```

Change line 50 to:

```
      50 LET a$(p)=n$,a$(p,17 TO)=USING$("####",lnum),p=p+1
```

```
Add: 65 SORT a$( TO p-1)
     66 FOR n=1 TO p-1: PRINT a$(n): NEXT n
```

If your membership list is an array, you can SORT it - but if it has entries like:

```
   Fred Bloggs, 24 Wyche Ave
   D. Munns, 69 Dross Rd.
```

you have a problem. You have to SORT according to some column, such as the one with "F" and "D" in, but this will not give a sensible order of surnames. You need to ensure all the data you want to SORT is neatly lined up, by entering it as e.g.:

```
   Bloggs Fred, 24 Wyche Ave
   Munns D., 69 Dross Rd.
```

Another way is to reserve, say, the first 12 characters in the strings in your array for first names, the next 20 for surnames, and so on.

Dear Andy,

My Microdrive cartridges are getting a bit worn now, and I have recently had a bit of bother with "losing" programs on a couple of cartridges... Is there a "Microdrive Doctor" program available which allows cartridges to be examined in detail to recover damaged or deleted files?.. If you know of anything in that line, perhaps you could publicise its existence... there could be other readers who would also appreciate it.

George Baldwin, 15 Oakley Close, Addlestone, Weybridge, KT15 2LT

I wrote some sort of sector recovery thingy when I last converted a game from Spectrum Microdrive to IBM PC (!) but when I looked at it again, it never returned the recovered sectors to a normal format, so it's not suitable. I am sure other readers would like to hear about any good products in this line.

Dear Dr. Wright,

I read with interest in No. 8 of the Newsletter about the 8056 printer driver software. I tried to obtain a copy of CRASH magazine's TECH TAPE which I believed held such an item - but alas I can't get hold of one as they have finished it. Can you or any of our "club" help me? I will be obliged.

Stephen Freeman, 22 Ford Drive, Yarnfield, Staffs. ST15 0RP

Can anyone help him? I can't.

Dear Andy,

Why is it when I try to use PROC SOUND (BB Newsletter no. 2) it resets my +2 to 48K mode? On the enclosed tape are two versions of a program I wrote based on Battleships the pencil and paper game. The first is a 128K Beta Basic version and the second is the original program I wrote way back in Dec. 1986 on a 48K Spectrum in Spectrum Basic. I think you will agree that the Mk II version is better (although whatever you do battleships is still a rather boring game).

N.V. French, Spalding, Lincs.

I thought your Battleships looked very professional - well done! Pity it is too long for the Newsletter. PROC SOUND may be crashing because you have over-written the start of BB's code. Find out what the variable RT is after BB's Basic has been merged, and make sure the loop at line 30 (BBN no. 2, p.3) does not POKE any addresses as high as RT. If it does, that is the problem, and the addresses in line 30 need to be lowered by (47070-rt), as does the USR in line 200 on page 5.

Dear Andy,

A line in a recent program had the form:

```
     ON m:
     IF C1 THEN S1:
     IF C2 THEN S2:
     IF C3 THEN S3:
     IF C4 THEN S4
```

where m=1 or 2 or 3 or 4, and S1-S4 are single unconditional statements. The consequent crazy results led me to deduce that, as far as ON is concerned, an IF statement is (at least) 2 statements... I had assumed that ON searched for colons to determine the prescribed statement, but it appears that it searches for command-keywords; by going through the line as it does when checking syntax?

Ettrick Thomson, Aldeburgh, Suffolk

*Not exactly. The Spectrum treats both colons and THENs as inter-statement markers, so there are actually 8 statements after ON in your example. (Try IF 1 THEN PRINT zzz - the error report is for statement TWO.) I used the ROM routine at 198BH to skip to the required statement. I don't think I'd like to try to alter the usual treatment of THEN, but your example could use ON m*2-1 to give the results you require.*

Dear Dr. Wright,

I always put the name of a defined procedure in reverse video... When I LLIST, I lose the reverse video feature. I wonder if you could supply a set of POKEs to send a desired control code sequence to the printer to correspond to the inv. video and true video characters found in a program listing? My printer has a code for inverse print: CHR$ 27;CHR$ 126;CHR$ 50;CHR$ 1... Other people may not have the reverse print facility on their printer, but may want to use the codes for underline (or italic, or whatever) instead.

Francesco Stajano, Rome

How about ALTER (CHR$ 20+CHR$ 1) (i.e. the control characters for INVERSE 1 that are put in a listing when you press the INV VIDEO key) TO (the codes you want)? And similarly for (CHR$ 20+CHR$ 0) (TRUE VIDEO). Then LLIST, then change everything back? It won't work for me because I need a "b" type channel for control codes to work, and that means I lose auto-line-feed with carriage return, unless I open the printer and reset some switches...

HIS REPLY:

I tried it, and it worked fine. Here is a procedure that implements your idea:

```
DEF PROC hardlist first,last
  DEFAULT first=1,last=9999
  ALTER (CHR$ 20) TO (CHR$ 27+CHR$ 126+CHR$ 50)
  KEYIN "LLIST first TO last"
  ALTER (CHR$ 27+CHR$ 126+CHR$ 50) TO (CHR$ 20)
END PROC
```

The KEYIN is used to prevent a failure in RENUM.

****
LAST WORD

My attempts to use direct photocopying of some submissions in issue 13 went badly wrong for the items from John Watkins on pages 8 and 9, for the first 150 newsletters, at least. The contrast on the originals was just too low and the listings are hard to read. I re-typed them for later copies. If you want a better copy of the offending page, just send me an S.A.E. (or even just an addressed envelope). Parts that were particularly hard to read where line 6100 which finishes: max,dec,neg,expo,b$ Line 6170 which starts: IF CODE a$>47 AND CODE a$<58 AND LEN b$<length THEN and Line 6180 which includes: CHR$ 8;" ";CHR$ 8; CHR$ 8. A slight flaw in the original program means that line 6220 should have brackets round: a$="E" OR a$="e" if extra "e"s are to be detected when entering numbers in exponent form. (I often have trouble with brackets myself!)

As usual, any Newsletter contributions will be greatly appreciated. Best wishes to you all till next issue - which I hope will not be as late as this one!

****
BB NEWSLETTER, 24 WYCHE AVENUE, KINGS HEATH, BIRMINGHAM, B14 6LQ